Faithfulness in Model-Based Development

Paul Caspi
Laboratoire (CNRS-UJF-INPG)

1. Model-based development in computer science and in control and how to make them converge
2. The case of sampling
3. A topological approach

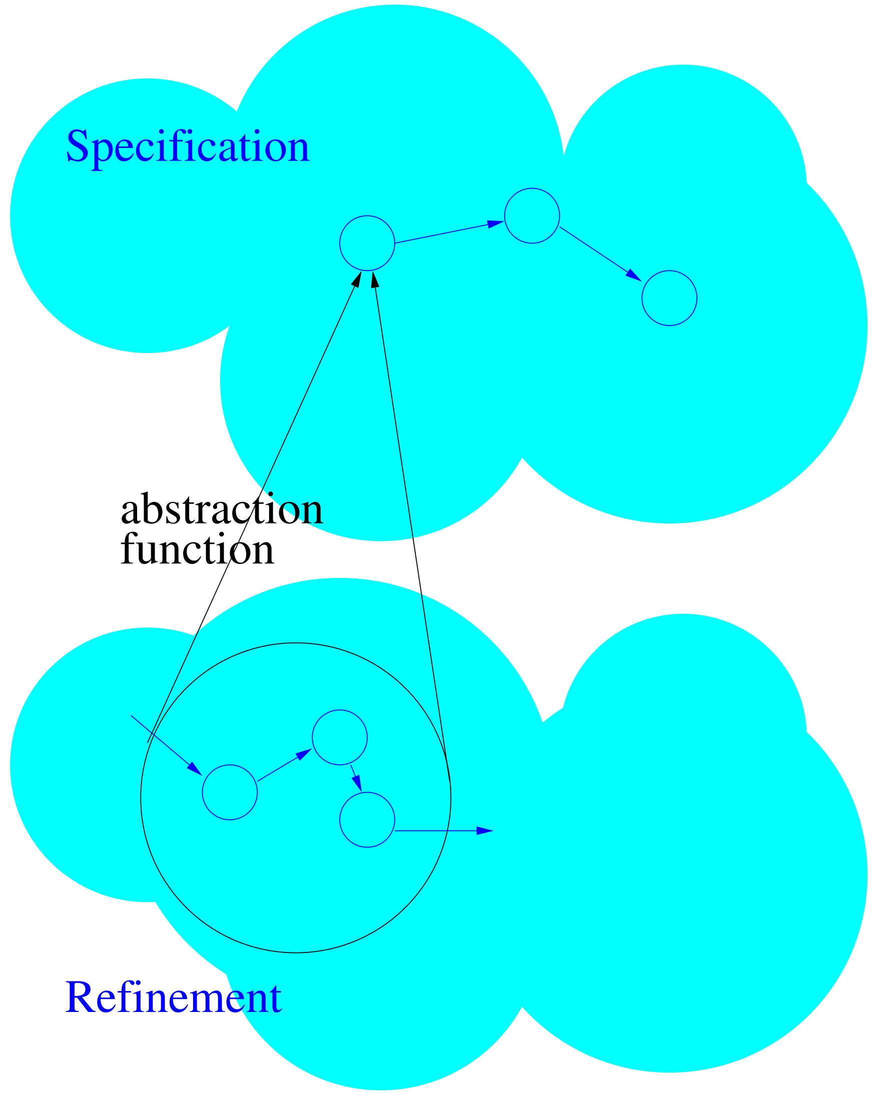
Model-based Design in Computer Science

- Starts from a non deterministic specification
- Based on successive property-preserving refinements
- Until an implementation is reached

Remarks:

- This is an idealised scheme, seldom fulfilled
- Yet has a paradigmatic value
- Some real-world impressive achievements in control!!
  - B method (Abrial): Paris, Barcelona, New York subways

Model-based design in computer science



B Example

$$
\begin{array}{ll}
MACHINE & initial \\
SETS & persons, buildings \\
ABSTRACT\_CONSTANTS & state, authorisation \\
PROPERTIES & \qquad persons \neq \emptyset \ \wedge \ buildings \neq \emptyset \\
 & \ \wedge \quad state \in persons \rightarrow buildings \\
 & \ \wedge \quad authorisation \in persons \leftrightarrow buildings \\
INVARIANT & state \subseteq authorisation \\
OPERATION & move \mathrel{\hat{=}} \begin{array}{ll} ANY & (p, b) \\ WHERE & (p, b) \in authorisation \\ & \wedge\ state(p) \neq b \\ THEN & state(p) := b \\ END & \end{array} \\
END &
\end{array}
$$

Model-based design in computer science

Further steps:

- Add implementation details:
  - paths, doors, badge controls,...
- Separate controllers from environment !!!
- Generate control programs

Model-based design control

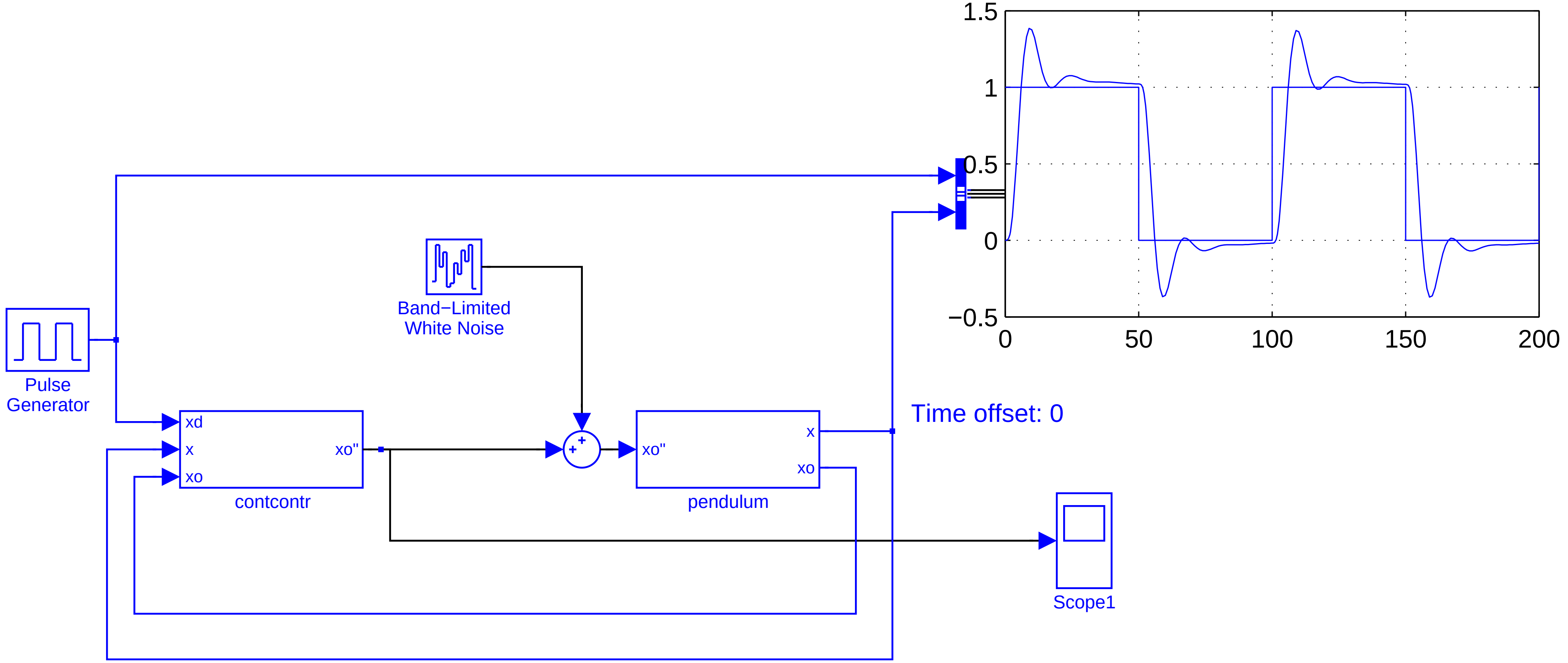
- Start from a perfect model
- Design a robust controller
- Add perturbations and implementation details and checks for robustness

Remarks:

- This is also an idealised scheme

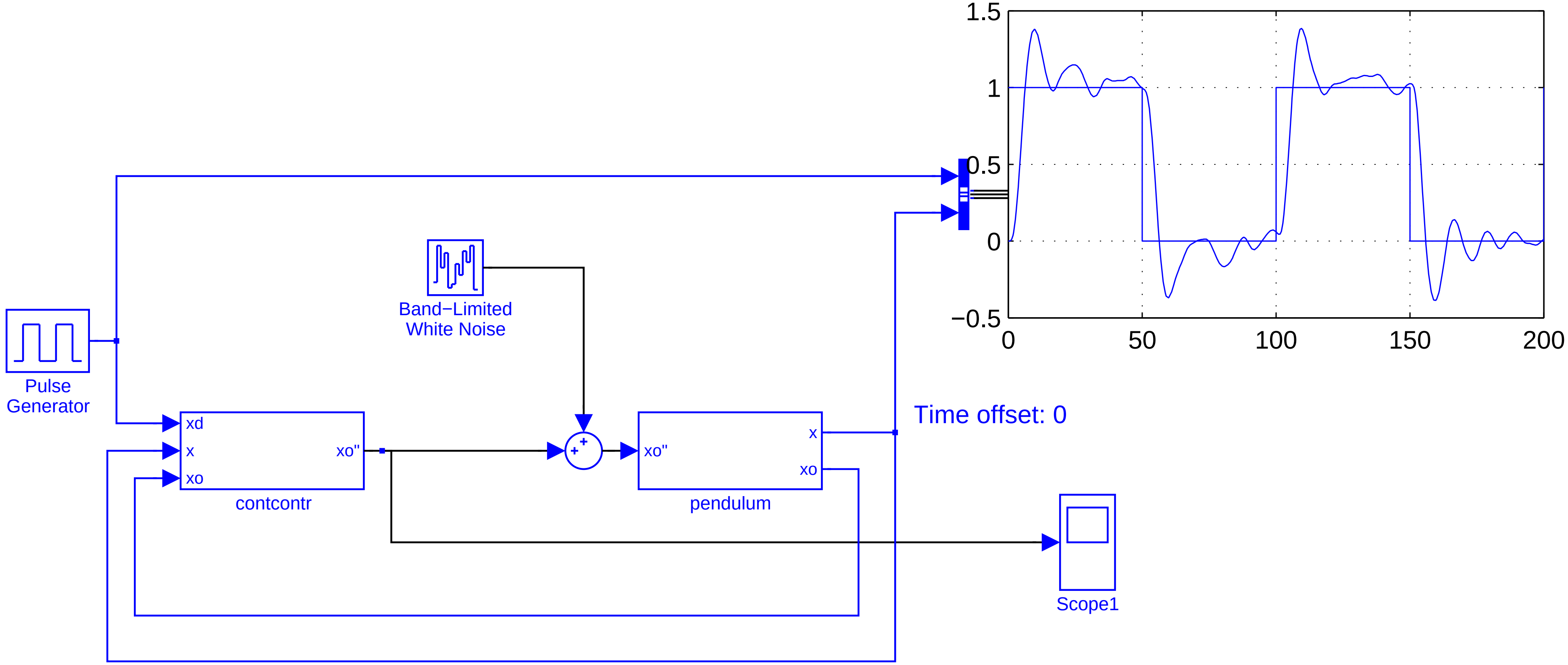
Model-based design in control

Perfect model



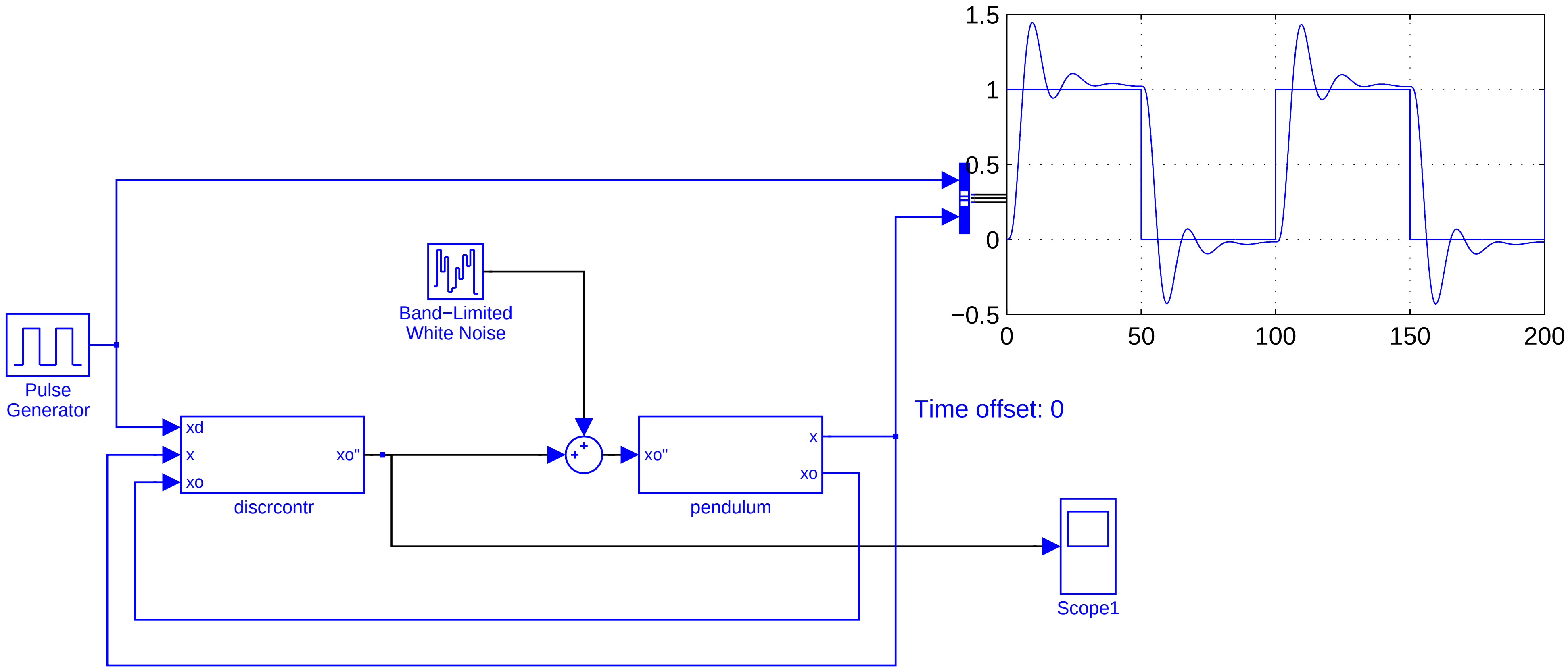
Model-based design in control

Perfect model with noise



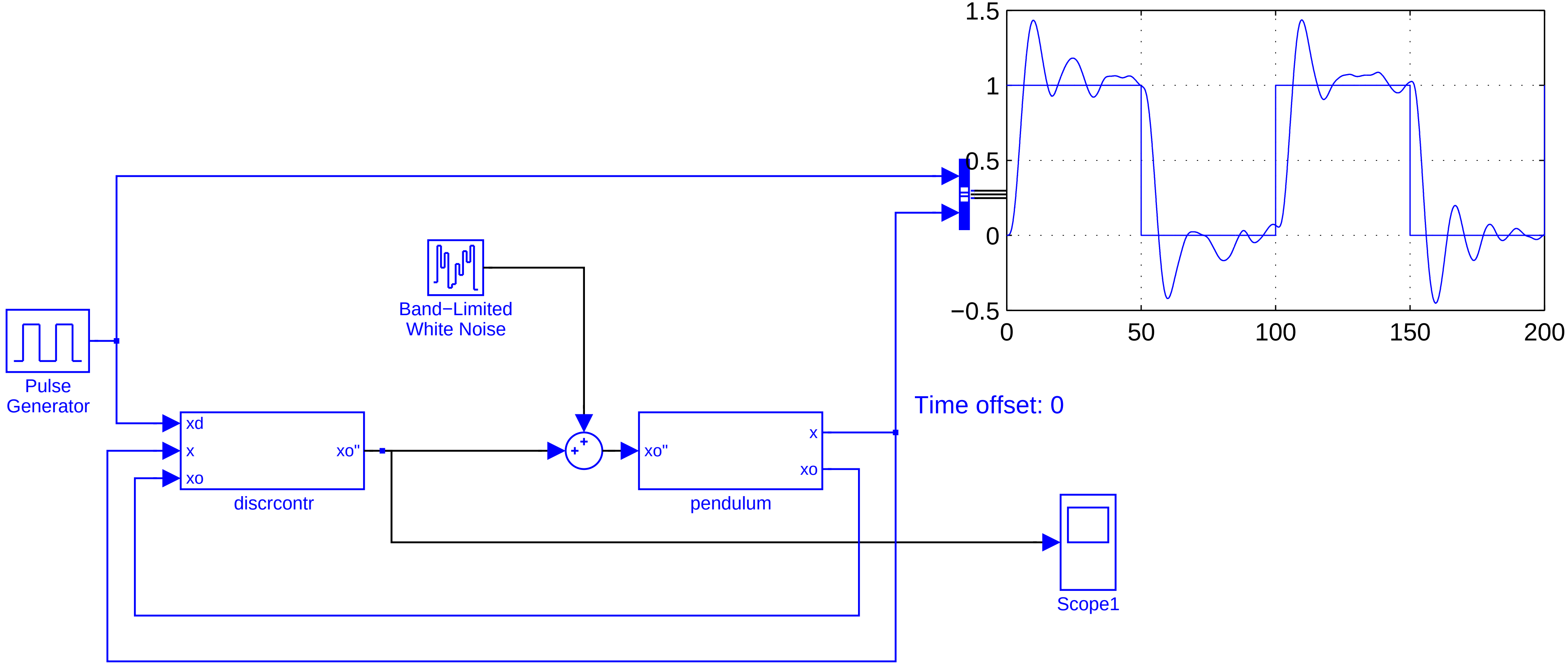
Model-based design in control

Discrete-time controller



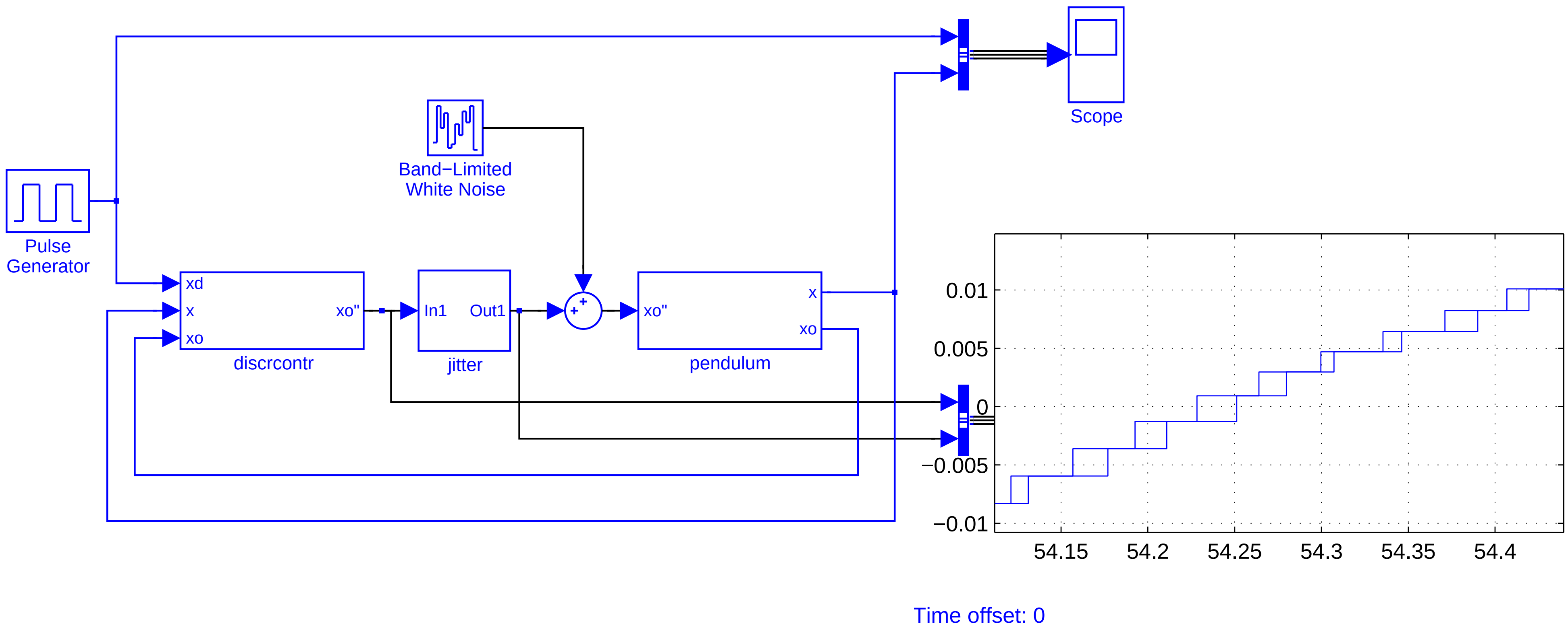
Model-based design in control

Discrete-time controller with noise



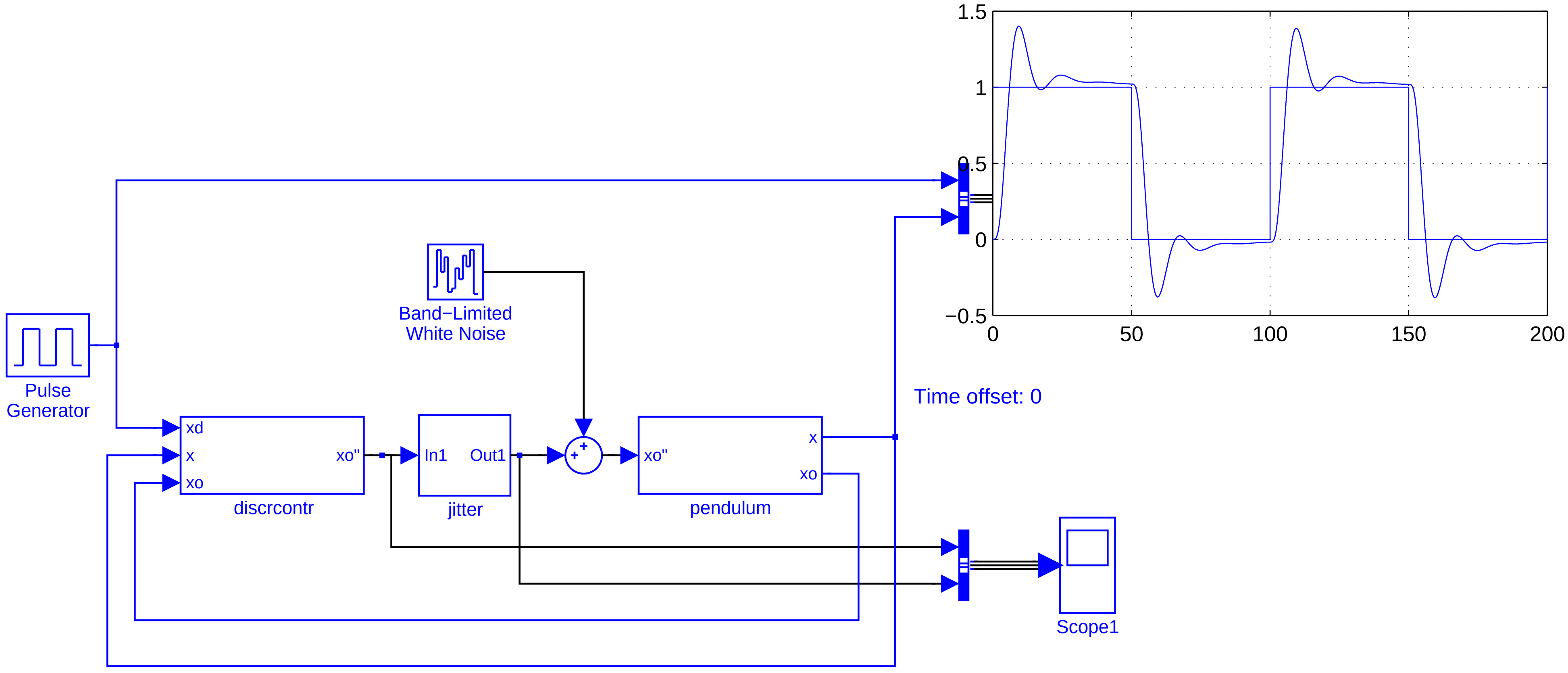
Model-based design in control

Discrete-time controller with jitter



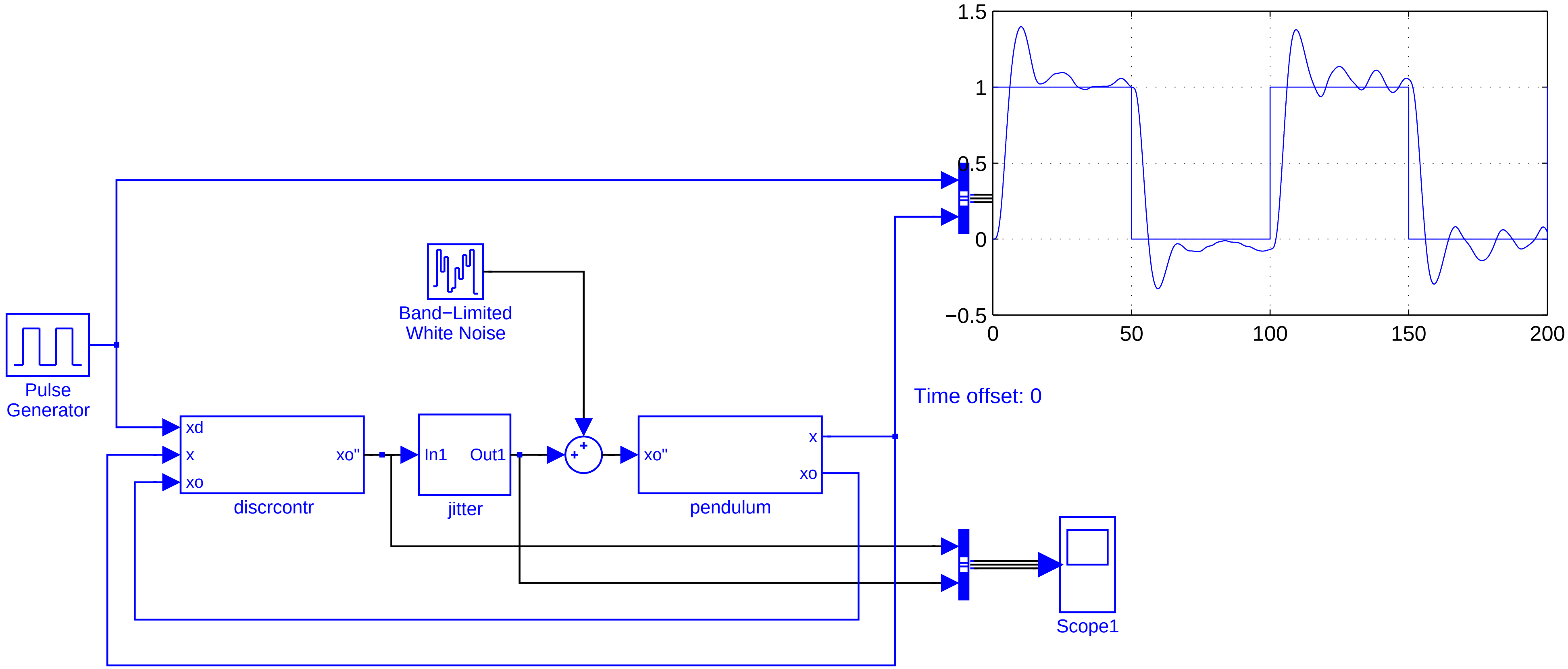
Model-based design in control

Discrete-time controller with jitter



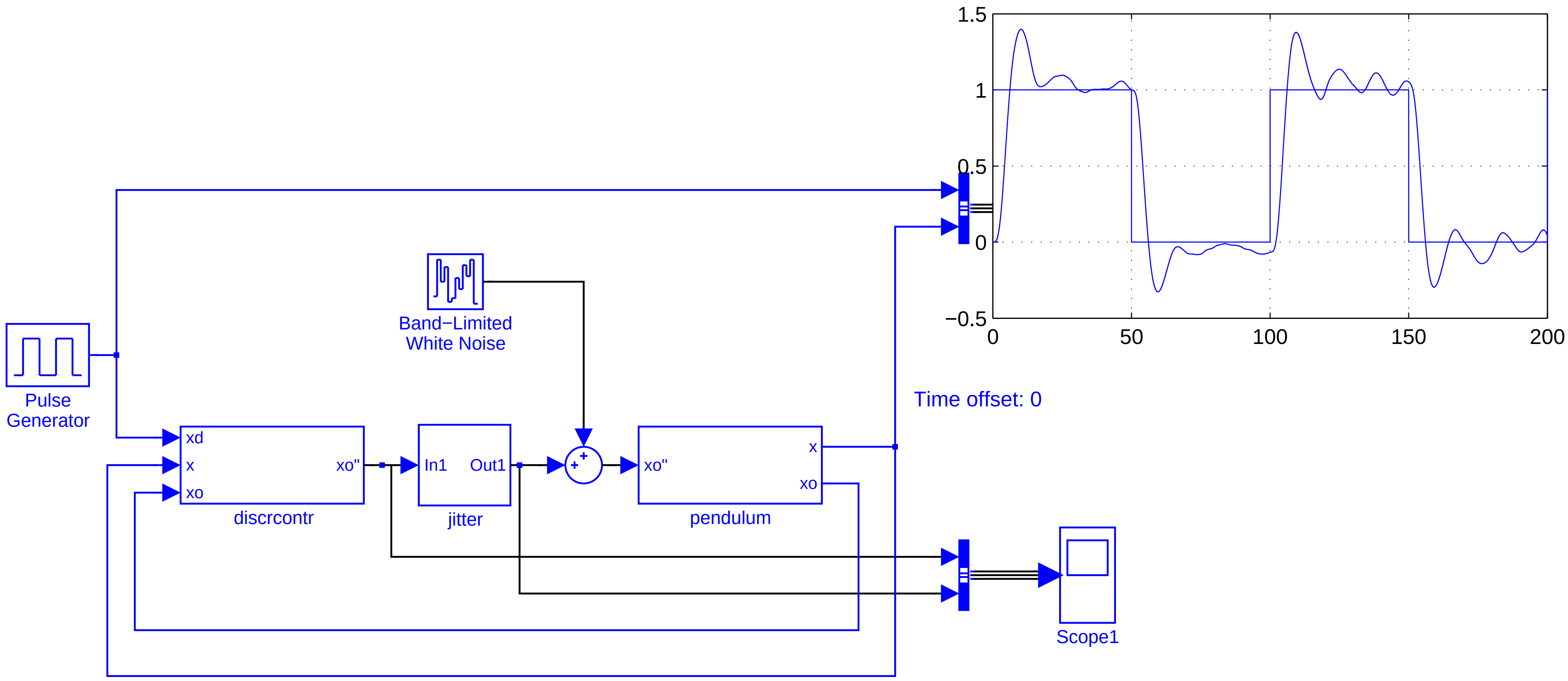
Model-based design in control

Discrete-time controller with jitter and noise



Model-based design in control

Discrete-time controller with jitter and noise



Is this enough? When should we stop adding implementation details?

A Possible Answer

Make computer science and control science converge ??

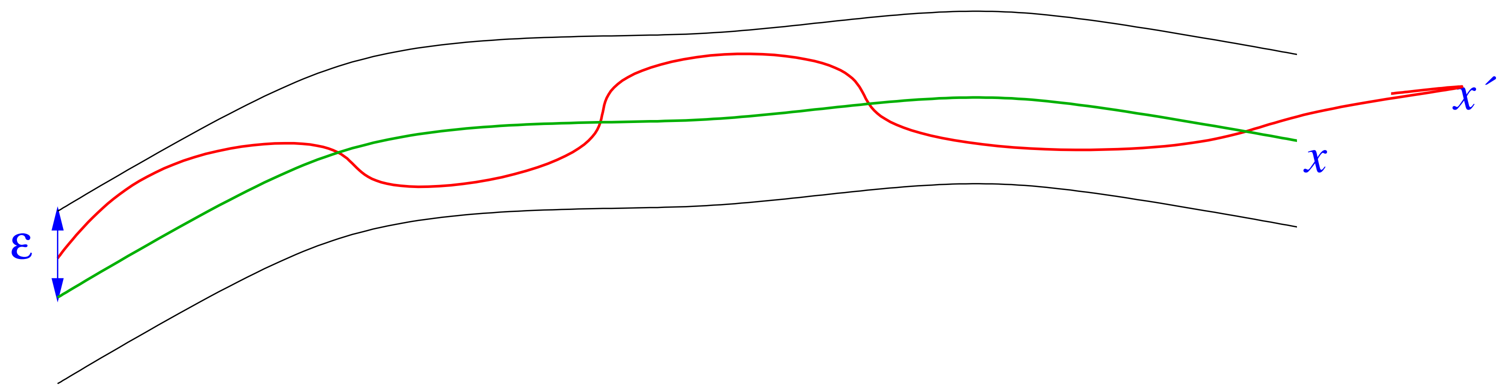
A suggestion :

Consider the perfect control model as specifying a set of behaviours, those behaviours which are within some “distance” of the perfect model behaviour.

This requires some notion of “distance”, able to account for

- perturbations
- modelling errors
- sampling
- jitter and communication delays
- priorities, distribution, ...

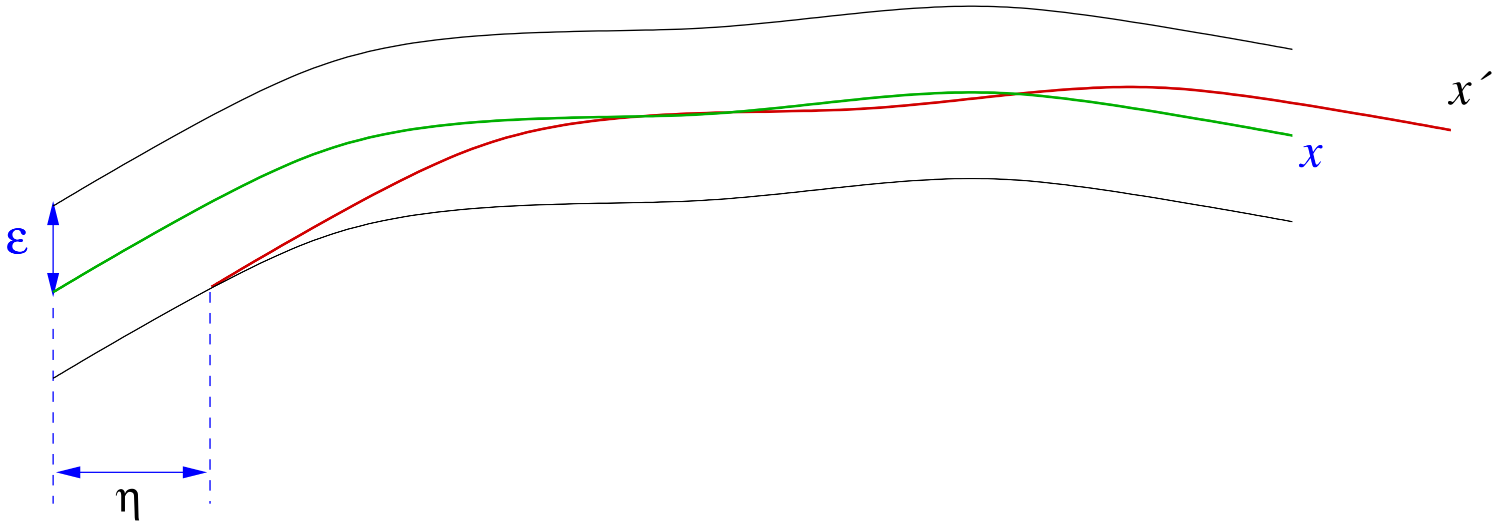
L_∞ Distance



$$||x - x'||_\infty \leq \epsilon$$

accounts for bounded modelling errors and pertubations

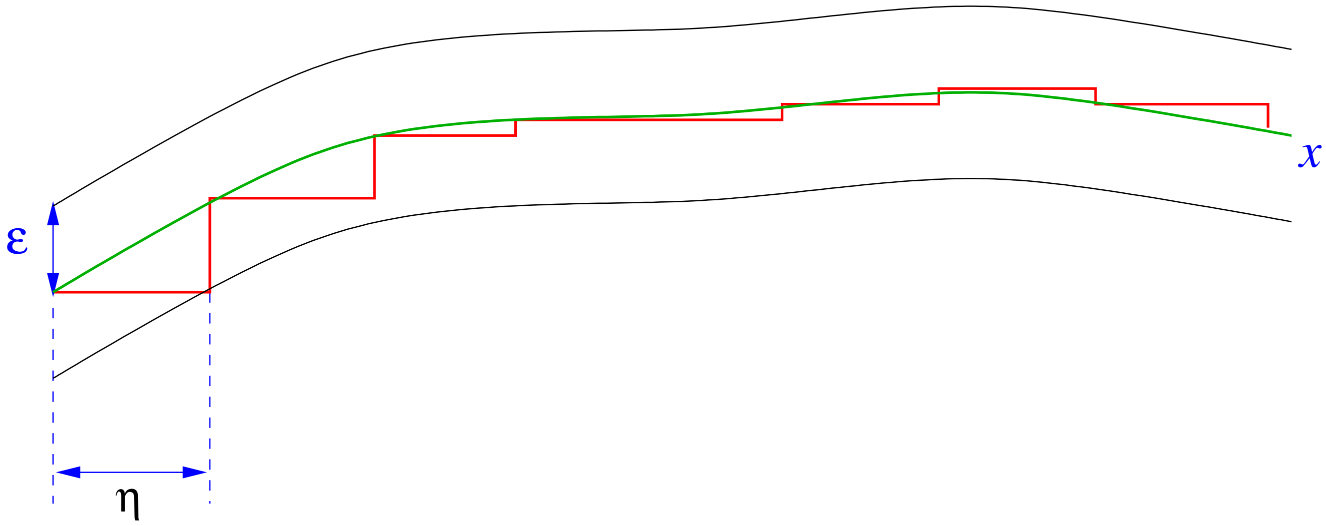
L_∞ Distance



$$\begin{cases} \tau \leq \eta \Rightarrow ||x - \Delta^\tau x||_\infty \leq \epsilon \\ \text{where } \Delta^\tau x(t) = x(t - \tau) \end{cases}$$

accounts for bounded jitter and communication delays

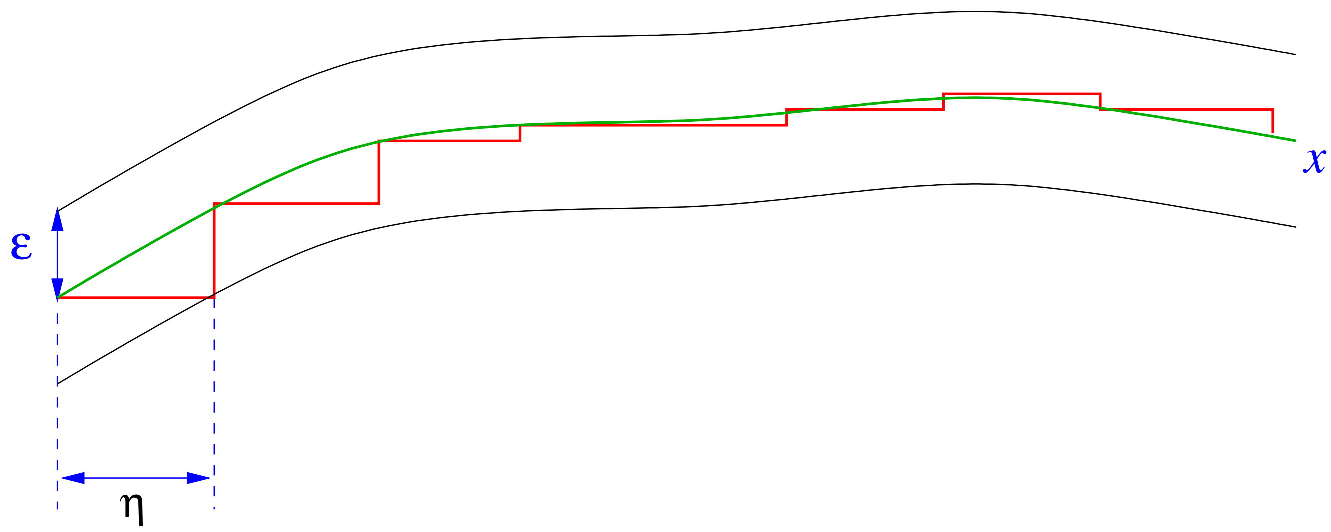
L_∞ Distance



$$\begin{cases} \tau \leq \eta \Rightarrow ||x - S_\tau x||_\infty \leq \epsilon \\ \text{where } S_\tau x(t) = x(\lfloor \frac{t}{\tau} \rfloor \tau) \end{cases}$$

accounts for periodic sampling

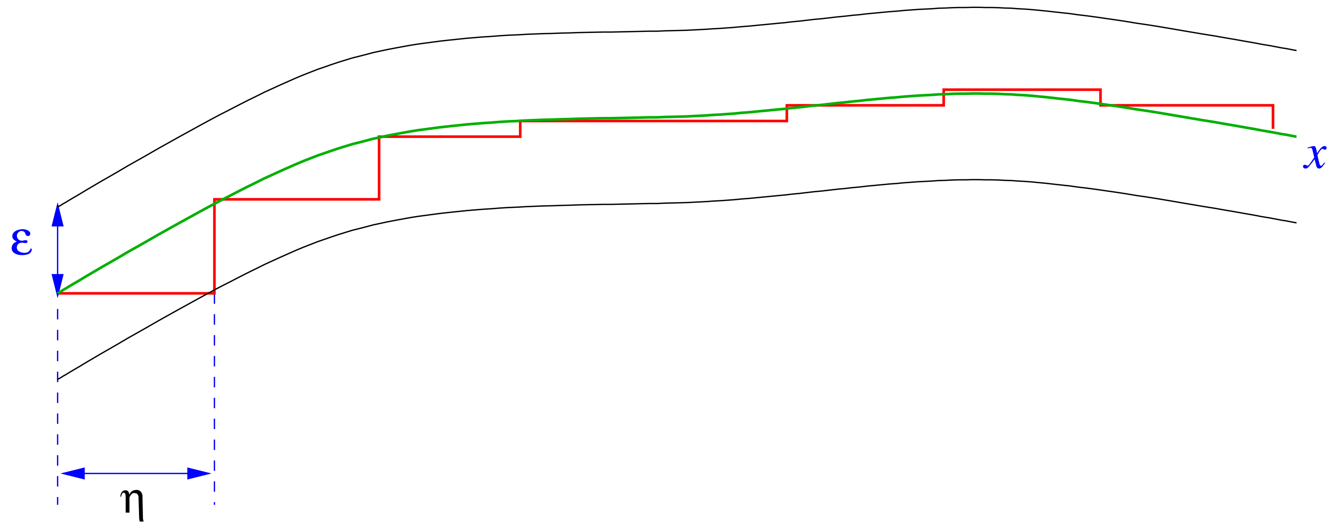
L_∞ Distance



$$\begin{cases} ||r - Id||_\infty \leq \eta \Rightarrow ||x - x \circ r||_\infty \leq \epsilon \\ \text{where} \quad Id(t) = t \\ \qquad\qquad f \circ g(t) = f(g(t)) \end{cases}$$

accounts for communication delays, jitter and sampling; r is called a retiming function

Delay-Error Function

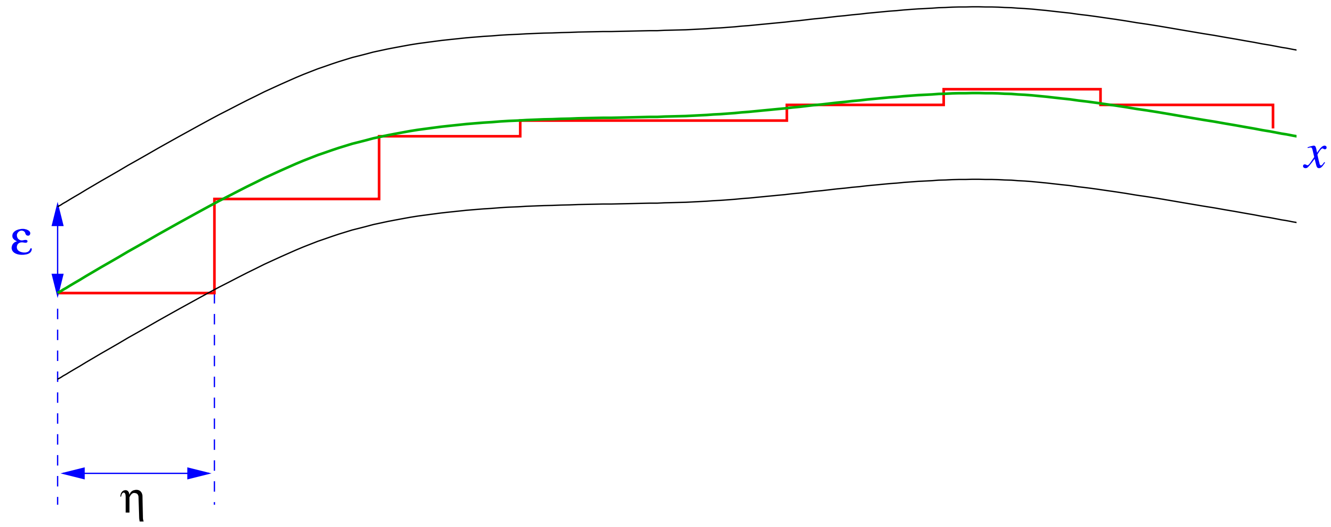


We can introduce the delay-error function η_x:

$$\begin{cases} \|r - Id\|_\infty \le \eta_x(\epsilon) \Rightarrow \|x - x \circ r\|_\infty \le \epsilon \\ \text{where} \quad Id(t) = t \\ \qquad\qquad f \circ g(t) = f(g(t)) \end{cases}$$

accounts for communication delays, jitter and sampling

Extension to Any Metric

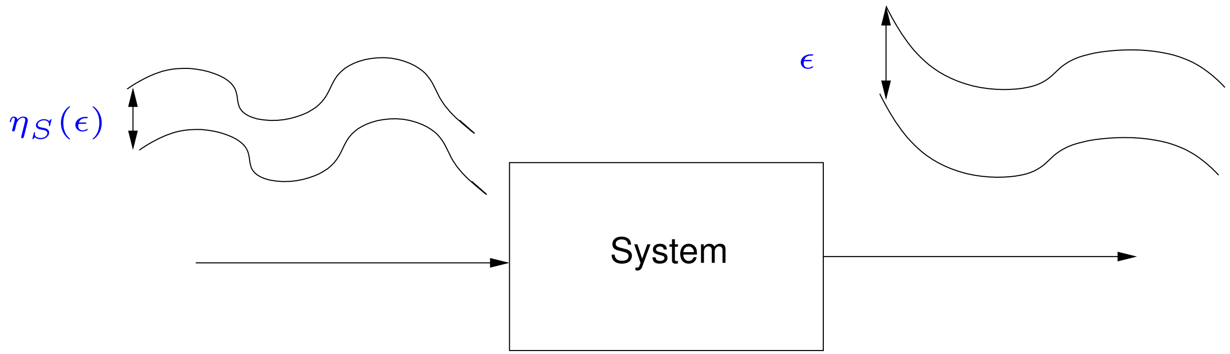


We can introduce the delay-error function η_x:

$$\begin{cases} ||r - Id||_\infty \leq \eta_x(\epsilon) \Rightarrow ||x - x \circ r|| \leq \epsilon \\ \text{where} \quad Id(t) = t \\ \qquad\qquad f \circ g(t) = f(g(t)) \end{cases}$$

Extension to Systems

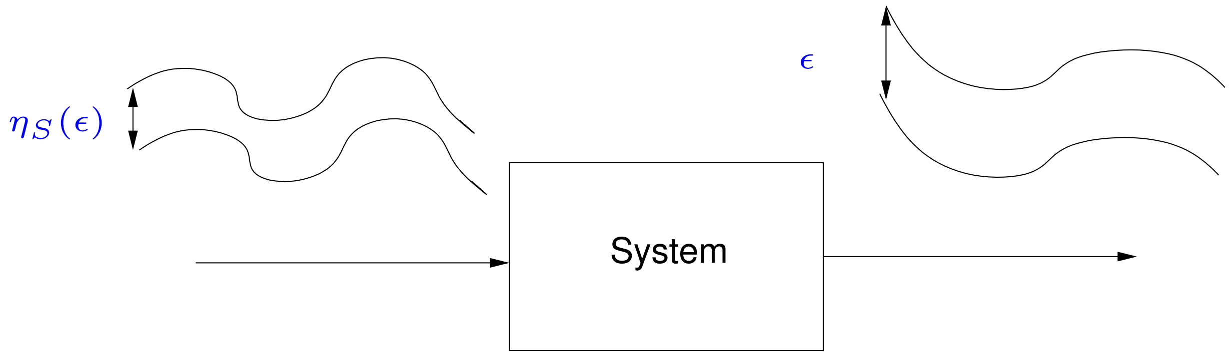
Uniformly Continuous Systems



$$\exists \eta_S > 0, \forall \epsilon > 0, \forall\ x, x' ||x - x'|| \leq \eta_S(\epsilon) \Rightarrow ||Sx - Sx'|| \leq \epsilon$$

Extension to Systems

Uniformly Continuous Systems



$$\exists \eta_S > 0, \forall \epsilon > 0, \forall\ x, x' ||x - x'||_\infty \leq \eta_S(\epsilon) \Rightarrow ||Sx - Sx'||_\infty \leq \epsilon$$

A UC time invariant system, fed with a UC input yields a UC output

Proof:

Given x UC, S UC, and $\epsilon > 0$, $\forall x'$,

$$||x - x'||_\infty \leq \eta_S(\epsilon) \Rightarrow ||(S\ x) - (S\ x')||_\infty \leq \epsilon$$

and $\forall \tau$,

$$|\tau| \leq \eta_x(\eta_S(\epsilon)) \Rightarrow ||x - (\Delta^\tau\ x)||_\infty \leq \eta_S(\epsilon)$$

Thus, $\forall \tau$,

$$|\tau| \leq \eta_x(\eta_S(\epsilon)) \Rightarrow ||(S\ x) - (S\ (\Delta^\tau\ x)||_\infty \leq \epsilon$$

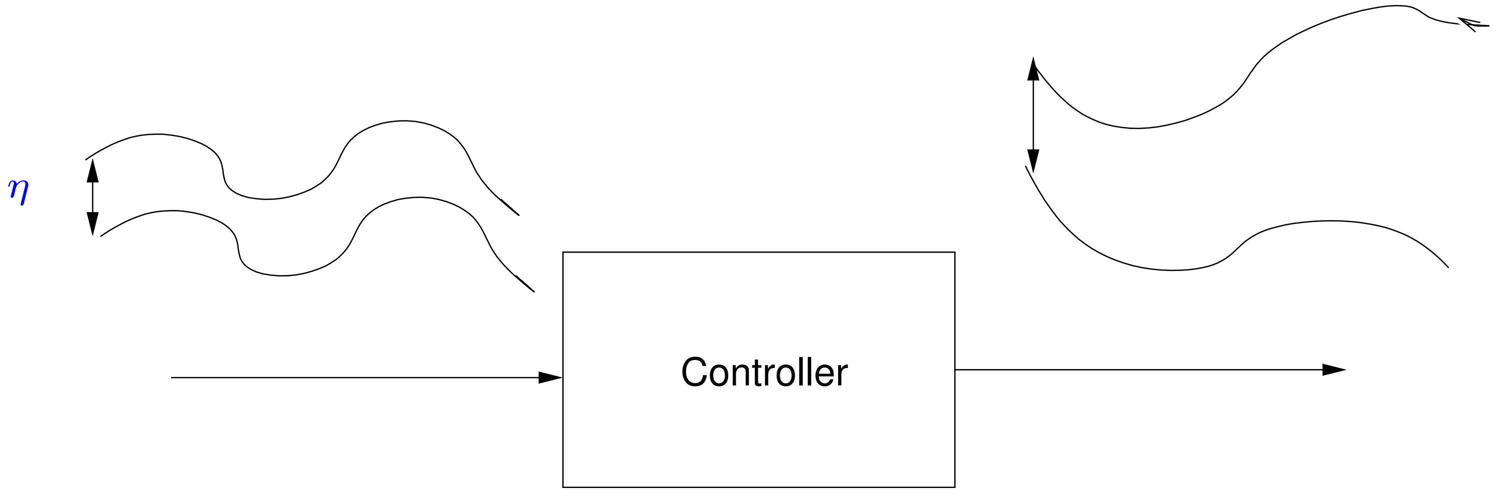
But $S(\Delta^\tau\ x) = \Delta^\tau(S\ x)$. We thus get

$$\eta_{Sx} = \eta_x \circ \eta_S$$

Extension to Systems

Actually, things are a bit more complicated as most systems are not UC:

Example : PID

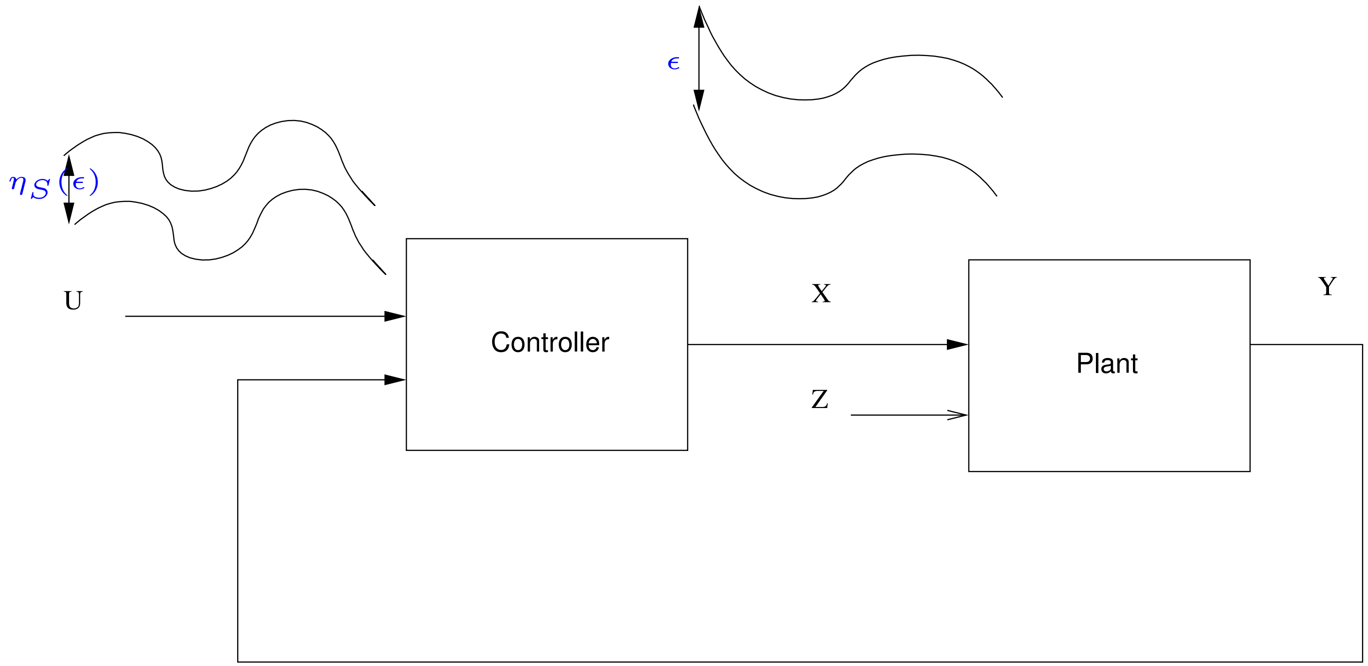


For instance unstable ones

Extension to Systems

Actually, things are a bit more complicated as most systems are not UC:

They become UC in closed loop with the environment



for instance an unstable controller can stabilise an unstable environment.

Conclusion

We need an approximation theory of embedded control systems in order to assess they robustness with respect to implementation details :

- Sampling
- Delays
- Priorities, distribution
- ...

Uniform continuity seems a good framework. Using an abstract functional approach allows us to paramtrise it with suitable metrics and even topologies.